A Debugging Approach for Trigger-Action Programming

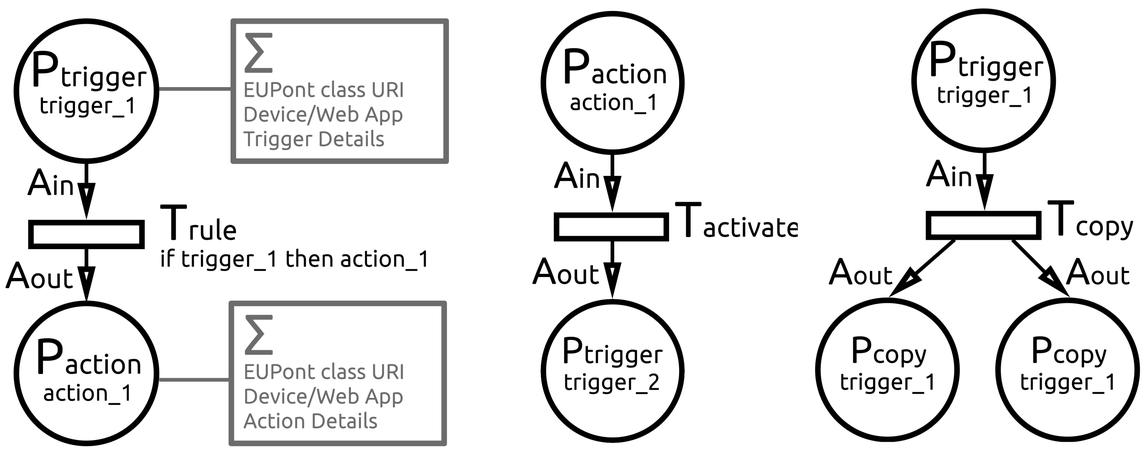
1 Motivation and Goal

End-User Development platforms, e.g., IFTTT, allow users to define their own trigger-action rules. Despite apparent simplicity, trigger-action programming may be a complex task, and errors can lead to unpredicted and dangerous behaviors, e.g., a door that is unexpectedly unlocked.

We present a novel debugging approach for trigger-action programming to assists end users at composition time. The goal is to properly warn users when they are defining any troublesome or potentially dangerous trigger-action rule, providing mechanism to understand why such problems can happen. Our approach is able to detect **loops**, **inconsistencies**, and **redundancies**.

2 SCPN Formalism

We propose the **SCPN** hybrid formalism, based on Petri Nets and Semantic Web, to model the execution behavior of trigger-action rules. Starting from a set of rules, **SCPN** uses a semantic model, named **EUPont**, to generate and analyze a Colored Petri Net. Triggers and Actions are modeled as **semantic places**, that can be connected together by means of **rule**, **activate**, or **copy transitions**.



3 Implementation

The defined rule could generate some problems

We integrated the **SCPN** formalism on top of IFTTT, one of the most popular EUD platforms. Users are informed of any loops, inconsistencies, or redundancies that the rule which is being defined may generate. By clicking an "explanation" button, users can inspect the problem by simulating the involved trigger-action rules step-by-step.

4 Preliminary Evaluation

Participants

Six students with different programming skills with a mean age of 20.00 years.

Methods

We asked participants to compose **12** trigger-action rules. When the EUD platform highlighted a problem, they could decide to save the rule or not. Before deciding, they could optionally use the explanation button to simulate the rules that generated the problem step-by-step.

Measures

As quantitative measures such as the number of saved and discarded rules, and the number of times participants used the explanation button.

Furthermore, when the participants decided to discard a rule, they had to explain the related problem (*explanation*). On the contrary, if they decided to save a problematic rule anyway, they had to motivate their choice (*justification*)

5 Results

	Loops	Inconsistencies	Redundancies
#	6	12	12
Discarded	5	12	5
Saved	1	0	7
Explanation	4	4	5

The debugging platform was successfully used in the study by all the participants. Quantitative measures suggest that:

- **loops** and **inconsistencies** are perceived as real problems;
- **redundancies** are perceived as less dangerous

Explanations and *justifications* suggest that:

- the **loop** is the most complex problem to understand

Open questions still remain to guide **future works**. How would end users actually debug their trigger-action rules, and which strategies would they adopt? To what extent they would benefit from our debugging approach with a large set of rules?

POLITECNICO DI TORINO
e-Lite

Luigi De Russis | Alberto Monge Roffarello

https://elite.polito.it